Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

MODEL PREDICTIVE CONTROL OF TIMED CONTINUOUS PETRI NETS

by
Huaiyu Zhan

This thesis addresses the optimal control problem of timed continuous Petri nets. The theory of Model Predictive Control (MPC) is first discussed. Then continuous Petri nets (PN) are introduced as a powerful tool for modelling, simulation and analysis of discrete event/continuous systems. Their useful capabilities are studied. Finally, a macroscopic model based on PN as a tool for designing control laws that improve the behavior of traffic systems is given. The goal is to find an approach that minimizes the total delay of cars in an intersection by computing the switching sequence of the traffic lights. The simulation results show that by using an MPC strategy to handle the variability of traffic conditions, the total delay is dramatically reduced.

MODEL PREDICTIVE CONTROL OF TIMED CONTINUOUS PETRI NETS

by
Huaiyu Zhan

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

January 2014

APPROVAL PAGE

MODEL PREDICTIVE CONTROL OF TIMED CONTINUOUS PETRI NETS

Huaiyu Zhan

Dr. MengChu Zhou, Thesis Advisor — Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. Yun-Qing Shi, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Mingming Yan, Committee Member — Date
Assistant Professor of School of Mechatronics Engineering, University of Electronic Science and Technology, Chengdu, China
Visiting Scholar, Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Huaiyu Zhan

Degree: Master of Science

Date: January 2014

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2014

- Bachelor of Science in Optoelectronics,
  Tianjin University, Tianjin, P. R. China, 2009

Major: Electrical Engineering

Because you never failed to support me in all my endeavors, I never failed to achieve success in my life. Thank you very much, Mom and Dad!

ACKNOWLEDGMENT

I would like to show my deepest appreciation to my thesis advisor, Dr. MengChu Zhou, a responsible and respectable professor, for his patience, kindness and enlightening guidance helped me all through my research.

Thanks to Dr. Yun-Qing Shi and Dr. Mingming Yan, for being the members of my thesis defense committee.

I shall extend my thanks to Ms. Sisi Li, a bright and obliging lady, for her helpful suggestion and insightful comments. My sincere thanks also go to Mr. Haobo Lai, who helps me a lot on my life and studies.

Last but not the least, my greatest appreciation goes to my parents Hanglun Zhan and Shirong Shen, for supporting me throughout my life.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Petri Nets (PN) are a mathematical formalism with a graphical feature usually and successfully used for modeling, analysis, synthesis and simulation of discrete event systems [1]. Discrete PN may suffer from a state explosion problem. Thus, one way to solve it is to use continuous Petri nets, which are considered in this thesis, offering a fluid approximation of the discrete event dynamics [2].

The basic idea of Model Predictive Control (MPC) is due to [3]: by optimizing a performance criterion over a future horizon, a sequence of control actions is computed and applied to the plant to be controlled.

This thesis proposes an optimal MPC strategy based on timed continuous Petri Nets. In particular, a traffic network model builds to study the advantages of this strategy.

In Chapter 2, MPC is introduced. Then in Chapter 3, the basic idea of continuous PN, and some useful models are analyzed. An intersection traffic model is then built and simulated with an MPC algorithm in Chapters 4 and 5, respectively. The conclusion and future work are given in Chapter 6.

The MPC of continuous PN related researches can be found in [4][5]. The traffic systems studies can be found in [6].

CHAPTER 2

MODEL PREDICTIVE CONTROL

2.1 Introduction

Model predictive control (MPC) has a long history in the field of control engineering. The general design objective of MPC is to compute a trajectory of a future manipulated variable u to optimize the future behavior of the plant output y. The terms are to be used frequently in following: the moving horizon window, prediction horizon, receding horizon control and control objective. They are introduced as below [7].

Moving horizon window: the time-dependent window from an arbitrary time t_i to $t_i + T_p$. Where t_i defines the beginning of the optimization window, and the length of the window T_p remains constant.

Prediction horizon: dictates how 'far' the future is predicted for. This parameter equals the length of the moving horizon window, T_p.

Receding horizon control: although the optimal trajectory of future control signal is completely described within the moving horizon window, the actual control input to the plant only takes the first sample of the control signal, while neglecting the rest of the trajectory.

In the planning process, the information at time t_i is in order to predict the future. This information is denoted as $x(t_i)$ which is a vector containing many relevant factors, and is either directly measured or estimated.

A given model that will describe the dynamics of the system is paramount in predictive control. A good dynamic model will give a consistent and accurate prediction of the future [8].

In order to make the best decision, a criterion is needed to reflect the objective. The objective is related to an error function based on the difference between the desired and the actual responses. This objective function is often called the cost function J, and the optimal control action is found by minimizing this cost function within the optimization window.

2.2 Single-input and Single-output System

Model predictive control systems are designed based on a mathematical model of a plant. The model to be used in the control system design is taken to be a state-space model. By using a state-space model, the current information required for predicting ahead is represented by the state variable at the current time. For simplicity, this work assumes that the underlying plant is a single-input and single-output system, described by:

$$x_m(k+1) = A_m x_m(k) + B_m u(k), \tag{2.1}$$

$$y(k) = C_m x_m(k), \tag{2.2}$$

where u is the manipulated variable or input variable; y is the process output; and x_m is the state variable vector with assumed dimension n_1.

Taking a difference operation on both sides of (2.1) and (2.2),

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k) \tag{2.3}$$

$$y(k+1) - y(k) = C_m A_m \Delta x_m(k) + C_m B_m \Delta x_m u(k). \tag{2.4}$$

Putting together (2.3) with (2.4) leads to the following state-space model:

$$\begin{bmatrix}\Delta x_m(k+1)\\ y(k+1)\end{bmatrix} = \begin{bmatrix}A_m & o_m^T\\ C_m A_m & 1\end{bmatrix}\begin{bmatrix}\Delta x_m(k)\\ y(k)\end{bmatrix} + \begin{bmatrix}B_m\\ C_m B_m\end{bmatrix}\Delta u(k)$$

$$y(k) = [o_m \quad 1]\begin{bmatrix}\Delta x_m(k)\\ y(k)\end{bmatrix}, \tag{2.5}$$

where $o_m = [0 \quad 0 \quad ... \quad 0]$. By letting $A = \begin{bmatrix}A_m & o_m^T\\ C_m A_m & 1\end{bmatrix}$, $B = \begin{bmatrix}B_m\\ C_m B_m\end{bmatrix}$, and $C = [o_m \quad 1]$, the triplet (A, B, C) is called the augmented model, which will be used in the design of predictive controllers.

2.3 Prediction of State and Output Variables

Upon the formulation of a mathematical model, the next step in the design of a predictive control system is to calculate the predicted plant output with the future control signal as the adjustable variables. Assuming that at the sampling instant $k_i > 0$, the state variable vector $x(k_i)$ provides the current plant information. The future control trajectory is denoted by

$$\Delta u(k_i), \Delta u(k_i + 1), \dots, \Delta u(k_i + N_c - 1),$$

where N_c is called the control horizon dictating the number of parameters used to capture the future control trajectory. The future state variables are

$$x(k_i + 1|k_i), x(k_i + 2|k_i), \dots, x(k_i + N_p|k_i),$$

where $x(k_i + m|k_i)$ is the predicted state variable at $k_i + m$. The control horizon N_c is less than (or equal to) the prediction horizon N_p.

Based on the state-space mode (A, B, C), the future state variables and the predicted output variables are, by substitution

$$x(k_i+1|k_i) = Ax(k_i) + B\Delta u(k_i)$$

$$\vdots$$

$$x\big(k_i+N_p\big|k_i\big) = A^{N_p}x(k_i) + A^{N_p-1}B\Delta u(k_i)$$

$$+\cdots+ A^{N_p-N_c}B\Delta u(k_i+N_c-1). \tag{2.6}$$

$$y(k_i+1|k_i) = CAx(k_i) + CB\Delta u(k_i)$$

$$\vdots$$

$$y\big(k_i+N_p\big|k_i\big) = CA^{N_p}x(k_i) + CA^{N_p-1}B\Delta u(k_i)$$

$$+\cdots+ CA^{N_p-N_c}B\Delta u(k_i+N_c-1). \tag{2.7}$$

Note that all predicted variables are formulated in terms of current state variable information $x(k_i)$ and the future control movement $\Delta u(k_i+j)$. Define vectors

$$Y = [y(k_i+1|k_i)\ \ y(k_i+2|k_i)\ \ \ldots\ y(k_i+N_p|k_i)]^T$$

$$\Delta U = [\Delta u(k_i)\ \ \Delta u(k_i+1)\ \ \ldots\ \Delta u(k_i+N_c-1)]^T.$$

Combine (2.6) and (2.7) together into a compact matrix form as

$$Y = Fx(k_i) + \Phi\Delta U, \tag{2.8}$$

where

$$F = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix};\ \Phi = \begin{bmatrix} \begin{pmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ CA^2B & CAB & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & \cdots & CA^{N_p-N_c}B \end{pmatrix} \end{bmatrix}$$

2.4 Optimization

For a given set-point signal $r(k_i)$ at sample time k_i, within a prediction horizon the objective of a predictive control system is to bring the predicted output as close as

possible to the set-point signal, where we assume that the set-point signal remains constant in the optimization window. This objective is then translated into a design to find the 'best' control parameter vector ΔU such that an error function between the set-point and the predicted output is minimized.

Assuming that the data vector that contains the set-point information is

$$R_a^T = [1 \quad \dots \quad 1]r(k_i),$$

the cost function J that reflects the control objective as

$$J = (R_s - Y)^T(R_s - Y) + \Delta U^T \bar{R} \Delta U, \tag{2.9}$$

where the first term is linked to the objective to minimize the errors between the predicted output and the set-point signal while the second term reflects the consideration given to the size of ΔU when the objective function J is made to be as small as possible. $\bar{\mathrm{R}}$ is a diagonal matrix in the form that $\bar{\mathrm{R}} = \mathrm{r_w I}$, where $\mathrm{r_w}$ is used as a tuning parameter for the desired closed-loop performance. For the case that $\mathrm{r_w} = 0$, the cost function (2.9) is interpreted as the situation that the goal would be solely to make the error $(\mathrm{R_s} - \mathrm{Y})^{\mathrm{T}}(\mathrm{R_s} - \mathrm{Y})$ as small as possible. For the case of large $\mathrm{r_w}$, the cost function (2.9) is interpreted as the situation that how large ΔU might be while cautiously reducing the error $(\mathrm{R_s} - \mathrm{Y})^{\mathrm{T}}(\mathrm{R_s} - \mathrm{Y})$.

To find the optimal ΔU that will minimize J, by using (2.8), J is expressed as

$$J = \big(R_s - Fx(k_i)\big)^T \big(R_s - Fx(k_i)\big) - 2\Delta U^T \Phi^T \big(R_s - Fx(k_i)\big) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U. \tag{2.10}$$

From the fist derivative of the cost function J:

$$\frac{\partial J}{\partial \Delta U} = -2\Phi^T \big(R_s - Fx(k_i)\big) + 2(\Phi^T \Phi + \bar{R}) \Delta U, \tag{2.11}$$

the necessary condition to minimize J is obtained as

$$\frac{\partial J}{\partial \Delta U} = 0,$$

from which the optimal solution for the control signal is

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1} \Phi^T \big(R_s - Fx(k_i)\big), \tag{2.12}$$

Note that R_s is a data vector that contains the set-point information expressed as

$$R_s = \overbrace{[1 \quad 1 \quad \dots \quad 1]}^{N_p}{}^T r(k_i) = \bar{R}_s r(k_i).$$

The optimal solution of the control signal is linked to the set-point signal $r(k_i)$ and the state variable $x(k_i)$ via the following equation:

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1} \Phi^T \big(\bar{R}_s r(k_i) - Fx(k_i)\big). \tag{2.13}$$

2.5 Receding Horizon Control

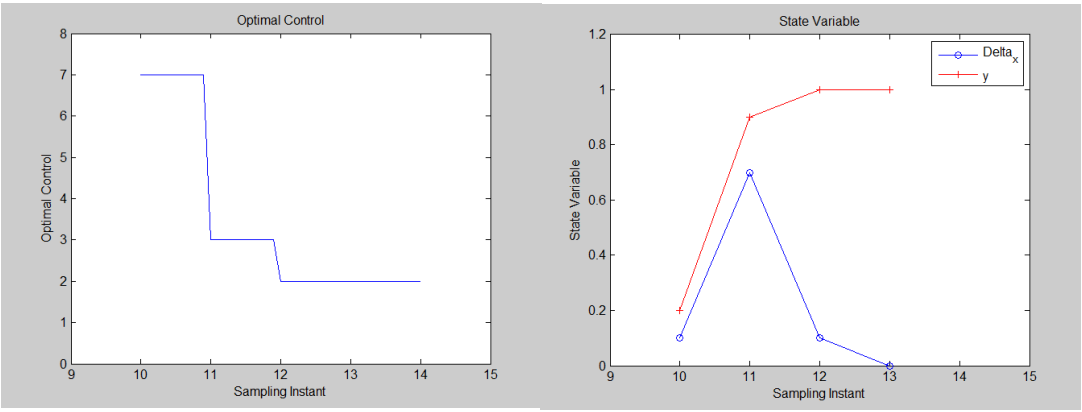

Fig. 2.1 Receding horizon control

Although the optimal parameter vector ΔU contains the controls $\Delta u(k_i), \Delta u(k_i + 1), \dots, \Delta u(k_i + N_c - 1)$, with the receding horizon control principle, we only implement the first one of this sequence, i.e., $\Delta u(k_i)$, while ignoring the rest of the sequence. When the next sample period arrives, the more recent measurement is taken to form the state

vector $x(k_i + 1)$ for calculation of the new sequence of control signal. This procedure is repeated in real time to give the receding horizon control law.

Fig. 2.1 shows the trajectories of the state variable Δx and y, as well as the control signal that is used to regulate the output. This example also illustrates the differences between the ΔU parameter vectors at different time instances. As the output response reaches the desired set-point signal, ΔU approaches zero.

CHAPTER 3

PETRI NETS

3.1 Introduction

In the engineering discipline, system evolution has invariably been facing three major needs.

1. The need to develop increasingly complex systems to accomplish more and more desired functions;
2. The need to assess the system's operational risks;
3. The need to have a cost competitive solution to attain these requirements.

Due to time and money constraints, it is no longer feasible to follow the design cycle of trial and error prototyping. Instead, the industry is leaning more towards simulation, such that the design flaws can be worked out even before the prototype is built.

That Petri nets (PN) comes in. They are a net-based abstraction, which can be used as a modeling tool (graphical and mathematical), as a simulation tool and as an analysis tool.

As a modeling tool, they helps one performs system design. The graphical nature aids system visualization while, the mathematical nature captures system behavior and allows one to perform a rigorous analysis of important properties.

As a simulation tool, they enables one to identify design errors. Extensive simulations may detect errors.

As an analysis tool, they can reveal various properties of the model and hence of the actual physical system. Thus, one can draw important conclusions about the system without going for experimentation or performing lengthy calculation of conventional system modeling.

3.2 Continuous Petri Nets

The PN systems that will be considered in this work are continuous. The basic difference between discrete and continuous PN (contPN) is that the components of the markings and firing count vectors are not restricted to take values in the set of natural numbers but can take non-negative real values. Let $\mathbb{R} > 0$ represent the set of all non-negative real numbers [5].

Definition 1: A continuous Petri net system is a pair $\langle N, m_0 \rangle$, where: $N = \langle P, T, Pre, Post \rangle$ is the net structure with set of places P, set of transitions T. $Pre\ and\ Post\ \in \mathbb{R}_{\geq 0}^{P \times T}$ are input and out functions that define the static structure of the net.

$m(\tau)$ is the marking at time τ and in discrete time $m(k)$ is the marking at sampling instant k, whre $\tau = k \cdot \Theta$ and Θ is the sampling period. m_i is the i-th component of marking m and represents the number of tokens in place p_i. The preset and postset of a node $x \in P \cup T$ are denoted $\cdot x$ and $x \cdot$, respectively.

A transition t in a continuous PN is enabled at m iff $\forall p_i \in \cdot t, m_i > 0$, and its enabling degree is

$$enab(t, m) = \min_{p_i \in \cdot t} \left\{ \frac{m_i}{Pre(p_i, t)} \right\} \tag{3.1}$$

An enabled transition t can fire in any real amount $0 \leq \alpha \leq enab(t, m)$ leading to a new marking $m' = m + \alpha C(P, t)$, where $C = Post - Pre$ is the token flow matrix. If m is reachable from m_0, through a sequence $\sigma = t_{r1}(\alpha_1)t_{r2}(\alpha_2) \dots t_{rk}(\alpha_k)$, the fundamental equation is:

$$m = m_0 + C \cdot \hat{\sigma}. \tag{3.2}$$

where $\hat{\sigma}$ represents the firing vector of σ and its i-th component means the number of firing times of transition t_i in σ.

Definition 2: A timed contPN system is a contPN system together with a vector $\lambda: T \rightarrow \mathbb{R}_{>0}$, where λ_j is the firing rate of t_j.

With infinite server semantics, the flow of transition t_j is given by:

$$f_j = \lambda_j enab(t_j, m) = \lambda_j \min_{p_i \in \cdot t} \left\{ \frac{m_i}{Pre(p_i, t)} \right\} \tag{3.3}$$

3.3 Capabilities of Continuous Petri Nets

To represent the behavior of some realistic models, this section analyzes some capabilities of continuous PNs with their fundamental structures.

Infinite server semantics is used in system models in which the processing speed, i.e., the flow of transitions, is proportional to the number of customers in the upstream place, i.e., proportional to the enabling degree. The following examples show how the flow of transitions and the rate of change of the marking of places can be affected by the arc weights.

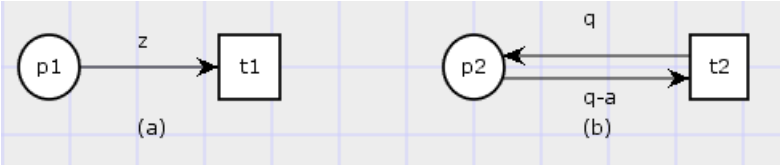

Fig. 3.1 In (a), the marking evolution does not depend on the arc weight. In (b), the marking evolution depends on the arc weights.

As in Fig. 3.1(a), consider transition t_1 has one input place p_1. Its flow is $f_{t1} = \lambda_{t1} \cdot m_{p1}/z$ where $z > 0$ is the weight of the arc. Under infinite server semantics the marking changes according to (3.2). Thus the evolution of the marking of p_1 does not depend on z, i.e., on the weight of the arc.

By slightly manipulating the system in Fig. 3.1 (a), it is possible to obtain a system in which the evolution op_1 depends on the weight of its input (output) arc. Consider the system in Fig. 3.1 (b), which is a self-loop. The flow of transition t_2 is $f_{t2} = \lambda_{t2} \cdot m_{p2}/q$, and the marking of p_2 depends on the parameter values q and a. If $a > 0$, the marking of p_2 decreases ($q > a$ is the condition). If $a = 0$, m_{p2} is constant; if $a < 0$, then m_{p2} increases.

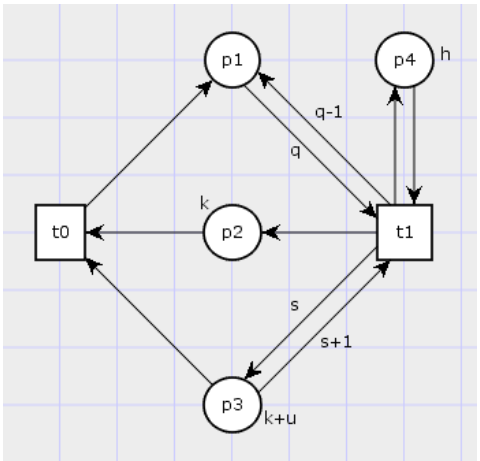

Fig. 3.2 Continuous PN with several self-loops

Following these ideas, the flow of a transition can be modeled as a piecewise linear function of the marking of a given place. As the model in Fig. 3.2, let the internal speed of t_1 be λ_{t1}, while the initial markings are given by $m_0(p_1) = 0, m_0(p_2) = k, m_0(p_3) = k + u$, and $m_0(p_4) = h$, where k, u, and h are positive real values. From the net structure, the following marking invariants can be deduced: $m(p_1) + m(p_2) = k, m(p_1) + m(p_3) = k + u, and\ m(p_4) = h$. It helps to synthesize the PN structure and to choose the arc weights that realize a given piecewise linear relationship between the marking $m(p_1)$ and the flow $f(t_1)$.

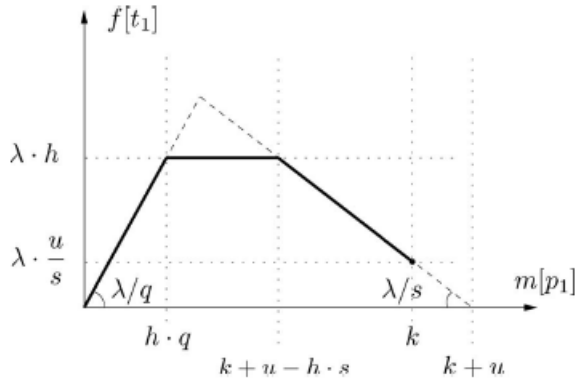

Fig. 3.3 Flow of transition t_1 is a piecewise linear function of the marking of p_1. Source: [11].

The line in Fig. 3.3 plots the piecewise linear relationship between $m(p_1)$ and $f(t_1)$. When the marking of p_1 is smaller than $h \cdot q$, it constrains the firing of t_1, and the flow of t_1 is proportional to $m(p_1)$. As soon as $m(p_1)$ satisfies $\frac{m(p_1)}{q} > h$, the flow of t_1 is constrained by p_4. Given that $m(p_4)$ is constant the flow will also remain constant. Assume that $m(p_1)$ keeps increasing. This fact involves a decrease in $m(p_2)$ and $m(p_3)$ since $m(p_1) + m(p_2) = k$ and $m(p_1) + m(p_3) = k + u$. Given that p_3 is also an input

place of t_1, it will constrain the flow of t_1 if $\frac{m(p_3)}{s} < h$ what is equivalent to $m(p_1) > k + u - h \cdot s$. Since all markings are positive and $m(p_1) + m(p_2) = k$, the maximum value that $m(p_1)$ can get is k. Summing up, the flow of t_1 is given by

$$f(t_1) = \begin{cases} \lambda(t_1) \cdot \frac{m(p_1)}{q}, & if\ m(p_1) < h \cdot q, \\ \lambda(t_1) \cdot h, & if\ h \cdot q \leq m(p_1) \leq k + u - h \cdot s, \\ \lambda(t_1) \cdot \frac{k+u-m(p_1)}{s}, & if\ k + u - h \cdot s < m(p_1). \end{cases}$$

Interestingly, the plot in Fig. 3.3 can be softened by adding more self-loops. Assume that a new place $p_5$is added to the net in Fig. 3 such that p_5 has the same arcs as p_1 but the arc weight is z, and $z > q$. Let $m_0(p_5) = m_0(p_5) + g$ be the initial marking of p_5. Clearly, when $m(p_1)$ is high enough, the firing of t_1 will be constrained by p_5. The new relationship between $m(p_1)$ and $f(t_1)$ is represented in Fig. 3.4.

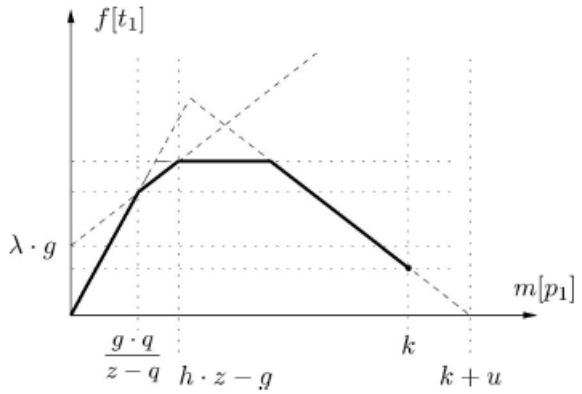

Fig. 3.4 Addition of a new self-loop can be used to slightly modify the piecewise relationship.

Source: [11].

This way, an appropriate choice of self-loop places and arc weights allows one to approximate any bell-shaped function, which is also known as the fundamental traffic diagram [12].

CHAPTER 4

TRAFFIC SYSTEM MODEL

4.1 Road Section

Petri nets can be used to design traffic control systems as done in [13] and [14]. This work builds a continuous Petri net traffic model. The model to be built requires a spatial discretization of the road, i.e., the road is divided into several sections [15].

Figure 4.1 shows the simple steps to model a given road section i. The number of cars in section i is represented by the marking of place p_1^i, the flow of cars leaving the section is the flow of transition t_i, and the flow of cars entering the section is the flow of transition t_{i-1}. If p_2^i and p_3^i are ignored, the use of infinite server semantics establishes $f(t_i) = \lambda(t_i) \cdot m(p_1^i)$, i.e., the outflow is proportional to the density. Hence, the subnet having p_1^i and t_i with an appropriate $\lambda(t_i)$ models free flow traffic. Notice that this relationship between the flow and marking $f(t_i) = \lambda(t_i) \cdot m(p_1^i)$ cannot be represented with finite server semantics where the flow of a transition is independent of the marking of its positively marked input places.

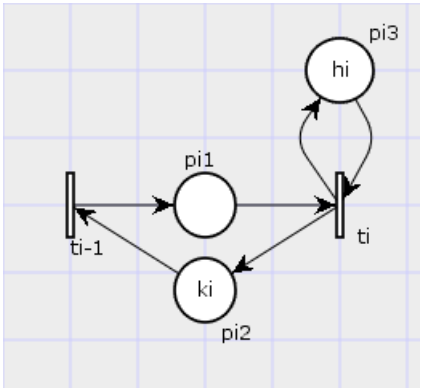

Fig. 4.1 PN model of a road section

For a better approximation of the *fundamental traffic diagram* [12], constant flow traffic can be modeled by adding $m(p_3^i)$. The marking of $m(p_3^i)$ is always constant and imposes an upper bound on the flow of t_i, $f(t_i) = \lambda(t_i) \cdot \min\{m(p_1^i), m(p_3^i)\}$.

Also, the model must impose an upper bound on the marking of the place representing the number of cars. $m(p_1^i) + m(p_2^i) = k^i$ ensures the capacity of the section, and $m(p_2^i)$ represents the gaps between cars.

4.2 Traffic Lights

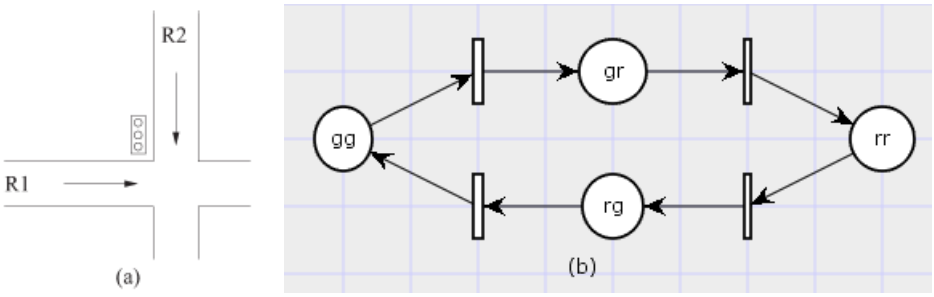

Fig. 4.2 Discrete PN modeling traffic lights in an intersection.
Source: [6].

Traffic lights can be seen as discrete event systems whose state can be either red, yellow, or green. This model as shown in Fig. 4.2 has four phases and only one token, the actions of each phase are:

1. gg: green lights for R1; and red lights for R2;
2. gr: yellow, then red light for R1; red, yellow, and finally green for R2;
3. rr: red lights for R1; and green lights for R2;
4. rg: red, yellow and finally green for R1, first yellow and then red for R2.

4.3 Intersection Model

Figure 4.3 shows how the flow of cars coming from Road 1 (R1) and Road 2 (R2) is regulated by the traffic light.

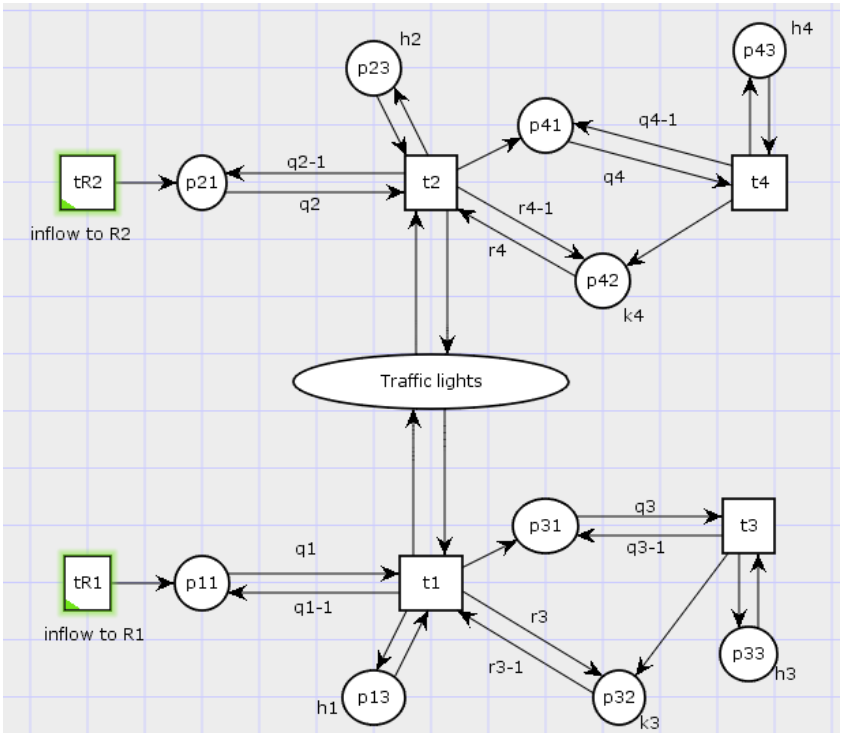

Fig. 4.3 Intersection modeled by a continuous PN

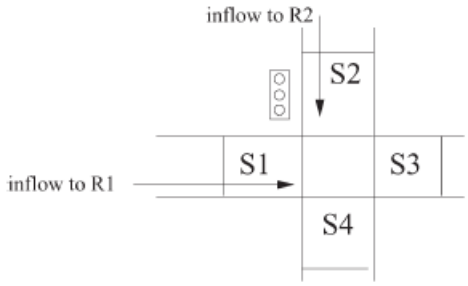

Fig. 4.4 Sketch of the traffic system in Fig. 8

The flow of cars crossing the intersection from R1 (R2) at a given time is obtained by multiplying the flow of the continuous transition at the corresponding time. For example, during phase gg the flow of transition t_1 is the same; but during phase gr, the flow decreases linearly and becomes zero after α time units; all along phase rr, the

flow remains zero; and finally, β time units before the end of rg, the flow increases linearly to original. Figure 4.4 shows the sketch of the traffic system in Figure 4.3.

4.4 Control Strategy

The control strategy used by the traffic lights is to minimize the sum of the time delays spent by all cars during a control horizon.

The total delay of all the cars passing through the system is obtained by summing over all the places p_1^i in the network

$$\sum_{p_1^i} \int_0^\rho m(p_1^i)(\tau)d\tau \tag{4.1}$$

Because the output transition of the given section in the traffic model is the input transition of the other section, (3.1) can be rewritten as the cost function

$$J = \sum_{p_1^i} \int_0^\rho m(p_1^i)(0)d\tau + \sum_{t\in T_i} \int_0^\rho \int_0^\tau f(t)d\xi d\tau - \sum_{t\in T_o} \int_0^\rho \int_0^\tau f(t)d\xi d\tau \tag{4.2}$$

where T_i and T_o represent the input (output) transitions of the system. In Figure 4.3, $T_i = \{t_{R1},\ t_{R2}\}$, and $T_o = \{t_3,\ t_4\}$.

To simplify the system, the first two terms of (4.2) do not depend on the control policy. It turns out that minimizing the total delay is equivalent to maximizing the outflow. Moreover, the flow of any output transition is piecewise constant.

At the end of the period H, where $H = \rho/\Delta$, $f^i(t)$ is the flow of transition t at the beginning of period i, the final expression for the cost function is

$$J' = max \sum_{i=1}^{H}((2\cdot(H-i)+1)\cdot\sum_{t\in T_o} f^i(t)). \tag{4.3}$$

In order to avoid very long waiting times for individual cars, a maximum time interval Red for red lights will be applied. Similarly, a minimum time interval $Green$ for green lights is used to avoid very short green light.

As all conditions show above, an MPC strategy can be given as follows:

$Algorithm$:

$Input$: $System\ structure, Initial\_state, H, Red, Green, Inflow\_cars$

1. $Current\_state := Initial\_state$
2. $loop$
    3. $Compute\ the\ potential\ sequences\ of\ traffic\ lights\ over\ control\ horizon\ "H"\ satisfying\ the\ "Red"\ and\ "Green"\ constraints$
    4. $Take\ the\ sequences\ \alpha\ that\ minimizes\ the\ total\ delay$
    5. $Apply\ the\ first\ control\ action\ \alpha\ on\ the\ traffic\ lights$
    6. $Get\ "New\_state"\ of\ the\ system\ after\ \Delta\ time\ units$
    7. $Curren\_state := New\_state$
8. $end\ loop$

The commands in steps 3 and 4 compute the control action for the next step according to the current state. The commands in steps 5, 6, and 7 apply the computed control action during one time period and update the system state. Given that the number of switching sequences is exponential with respect to the number of traffic lights, the computation time of step 4 might become too high if one must check every single sequence to find the optimal one. Fortunately, only the sequences satisfying the "Red" and "$Green$" constraints must be checked. The optimal sequence is obtained by simulation: after the simulation of each feasible sequence, the one yielding the minimum total delay is selected.

CHAPTER 5

SIMULATION

5.1 Model Parameters

This traffic system, modeled in Figure 4.3 and sketched in Figure 4.4, consists of two one-way streets Road 1 and Road 2 that cross at an intersection. Each road has two sections, which Road 1 has Section 1 and Section 3, and Road 2 has Section 2 and Section 4.

For simplicity, the model parameters for all sections are assumed to be the same, which the capacity of each section is 60 cars, and $q_1 = q_2 = q_3 = q_4 = 100$, $\lambda(t_1) = \lambda(t_2) = \lambda(t_5) = \lambda(t_6) = 4$, $\lambda(t_3) = \lambda(t_4) = 5$, $m(p_3^1) = m(p_3^2) = m(p_3^3) = m(p_3^4) = 0.4$. Set the initial distribution of cars in the system is $m_0(p_1^1) = 15$, $m_0(p_1^2) = 20$, $m_0(p_1^3) = 10$ and $m_0(p_1^4) = 15$. Thus $m_0(p_2^3) = 25$ and $m_0(p_2^4) = 45$, as the capacity is 60.

The flow rate of cars per second for Road 1 is set to be a uniform random variable distribution as $[0.2, 0.3]$, respectively, and for Road 2, $[0.4, 0.6]$. $\Delta = 8\ s$, $\alpha = 3\ s$ and $\beta = 2\ s$. The time of red lights is no longer than $6 \cdot \Delta = 48\ s$. The control horizon is set as 48 s.

5.2 Simulation Results

The goal of MPC is to minimize the total delay of cars in the intersection system. Fig. 5.1 shows the evolution of the distribution of cars in S1, S2, S3 and S4, where m1, m2, m3 and m4 stand for $m(p_1^1)$, $m(p_1^2)$, $m(p_1^3)$ and $m(p_1^4)$, respectively. The green stars represent the traffic light. R1 has green light, i.e., red light for R2 when stars in the figure are located at the value equal to 1. R1 has Red lights and R2 has green lights when stars are at 3. When green stars go to 2, the traffic light is switching.

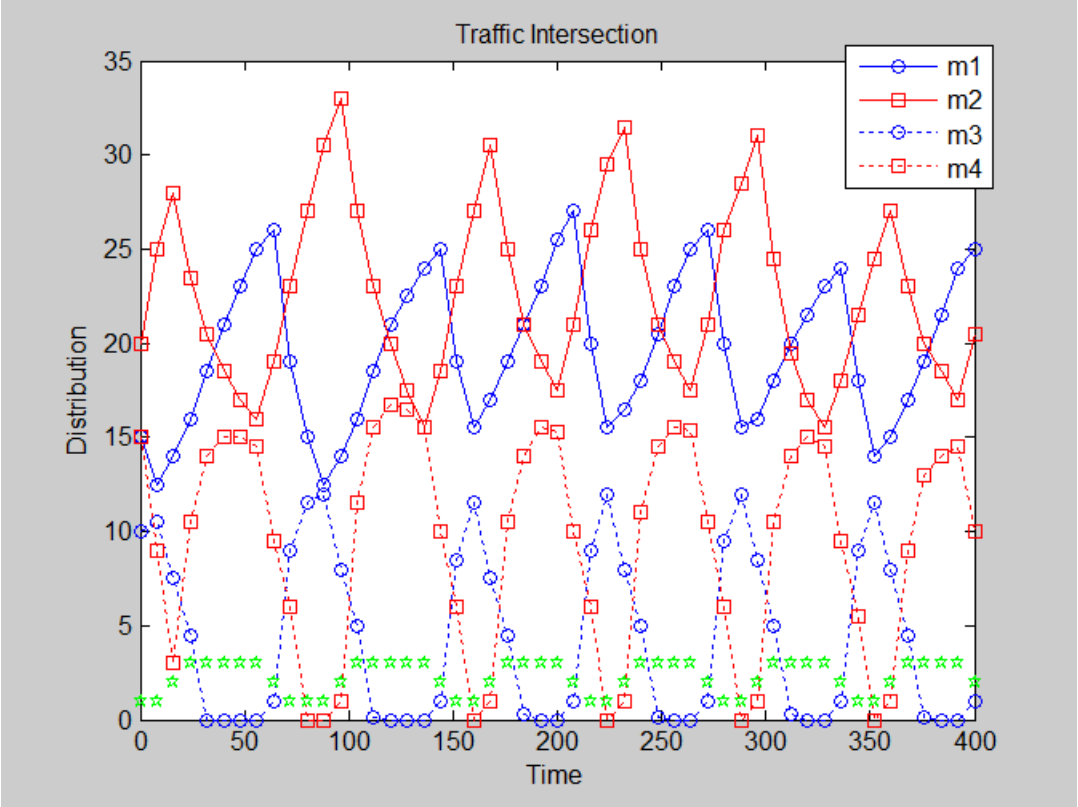

Fig. 5.1 Evolution of the systems in Fig 4.3 under MPC

The result of MPC shows that the time of green lights for R2 is longer than for R1, as the given car flow to R2 is bigger than the flow to R1. Because the inflow to R2 is a random variable, the time last or traffic lights is not constant.The number of cars in each section is never bigger than 35, showing that the intersection system is working properly.

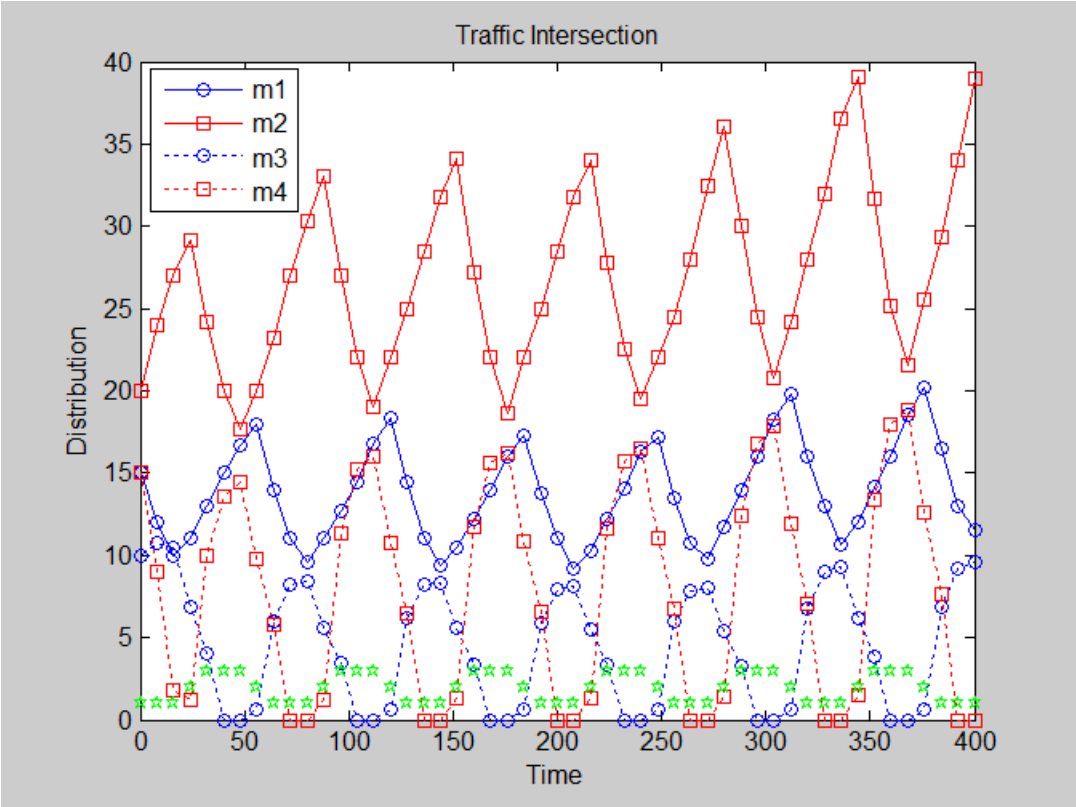

Fig. 5.2 Evolution of the system without MPC

Fig. 5.2 shows the evolution without MPC. A constant switching traffic light applies to the system, which is 32 s for green lights and another 32 s for red lights. Obviously, as the queue at m2 goes to longer and longer, and it will exceed the capacity of the road. Thus cars on R2 are going to suffer a traffic jam.

As an outstanding feature, the MPC can well react to any sudden traffic conditions change. Assume that at time $\tau = 200$, the flow into R2 changes from a random variable in [0.4, 0.6] to a constant rate of 0.3. Fig. 5.3 shows how the MPC adjusts the green ratio for R2 after the flow change. The green lights time for R2 is now reduced to 24 s, while all the cars cross the section share a low delay rate.

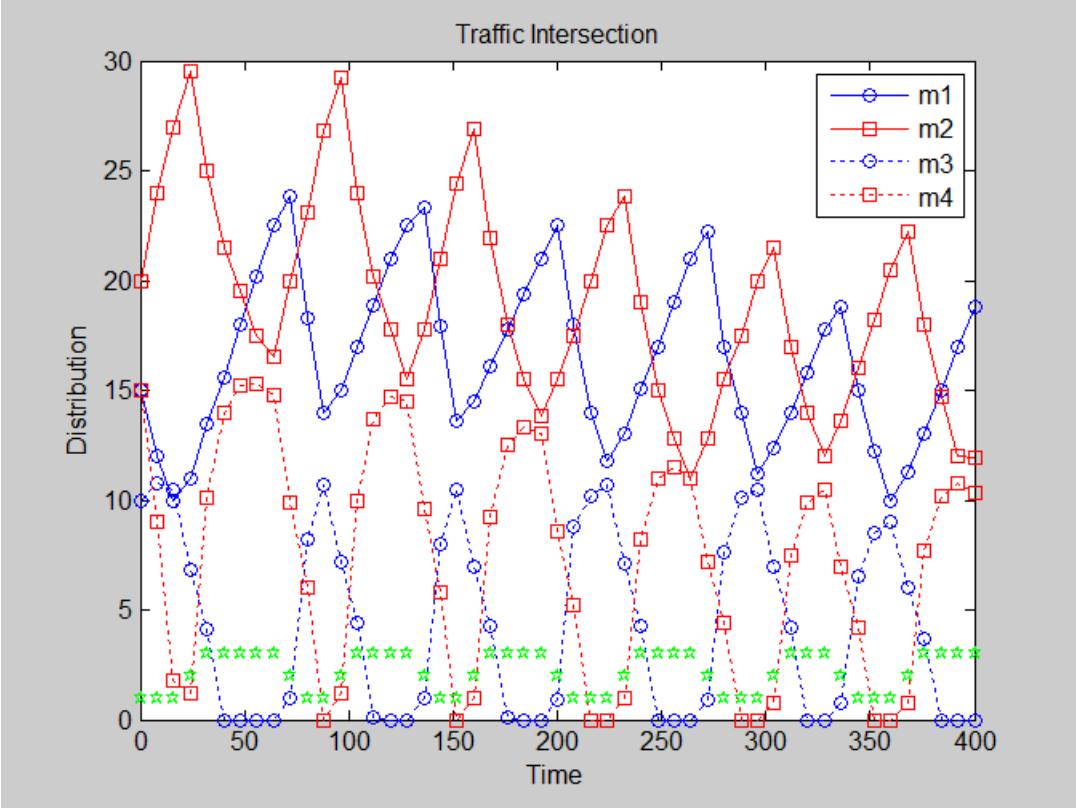

Fig. 5.3 MPC control of the system after a flow change

CHAPTER 6

CONCLUSION

The theory and applications of Model Predictive Control and timed continuous Petri Net are studied.

Based on timed continuous Petri Nets, an intersection traffic system model is built. In such a model, the marking of a place represents the number of cars in a given section, and the firing speed of its output transitions stands for the flow of cars leaving that section. The advantages of continuous PN are surely helpful, as it can model a traffic system properly, and the different parts of the system can be separately designed and easily assembled.

The control strategy of the traffic system is Model Predictive Control, which is a good choice to handle the changing traffic conditions. The simulation results show that this approach can minimize the total delay of cars in the traffic system remarkably.

On the other hand, the parameters here in this PN model and the strategy of MPC are largely chosen based on the idea of simplifying the question but not totally based on the original one. It only considers one intersection with two one-way road crossing, but not the total nearby traffic network. With a different model, the stability of the MPC based on continuous PN may become an issue. Hence much more works are required in the future.

REFERENCES

[1] M. Zhou and K. Venkatesh, *Modeling, Simulation,, and Control of Flexible Manufacturing Systems: A Petri Net Approach*, Singapore: WSPC, 1999.

[2] M. Silva and L. Recalde, "Petri Nets, and Integrity relaxations: A View of Continuous Petri Net Models," in *IEEE Trans on Systems, Man, and Cybernetics*, part C, Volume. 32, No. 4, pp.314-327, 2002.

[3] J. Richalet, A. Rault, J. L. Testud, and J. Papon, "Model Predictive Heuristic Control: Applications to Industrial Processes," in *Automatica*, Vol. 14, No. 5, pp. 413-428, 1978.

[4] A. Guia, C. Mahulea, L. Recalde, C. Seatzu, and M. Silva, "Optimal Control of Continuous Petri Nets via Model Predictive Control," in *8th International Workshop on Discrete Event Systems*, Ann Arbor, MI, 2006.

[5] A. Guia, C. Mahulea, L. Recalde, C. Seatzu, and M. Silva, "Properties of Continuous Petri Nets Controlled via Model Predictive Control," in *9th International Workshop on Discrete Event Systems*, Gothenburg, Sweden, 2008.

[6] J. L. Gallego, J. L. Farges, and J. J. Henry, "Design by Petri nets of an intersection signal controller," in *Transp. Res. Part C: Emerging TechNol.*, Vol4, No. 4, pp. 231-248, 1996.

[7] L. Wang, *Model Predictive Control System Design, and Implementation Using MATLAB*, London, UK: Springer-Verlag, 2009.

[8] E. FampoNogara, D. Jia, B. H. Krogh, and S. Talukdar, "Distributed Model Predictive Control," in *IEEE Control Systems*, Feb 2002.

[9] P. Thomas, C. Tolba, D. Lefebvre, and A. El Moudni, "Continuous and timed Petri nets for the macroscopic and microscopic traffic flow modelling," *Simul. Modelling Pract. Theory*, Vol. 13, No.5, pp. 407-436, Jul. 2005.

[10] M. Silva and L. Recalde, "On Fluidification of Petri Net Models: From Discrete to Hybrid, and Continuous Models," *in Annu. Rev. Control*, Vol. 28, No. 2, pp. 253-266, 2004.

[11] A. Di Febbraro, D. Giglio, and N. Sacco, "Urban Traffic Control Structure Based on Hybrid Petri Nets," in *IEEE Trans. Intel. Transp. Syst.*, Vol. 5. No. 4, pp. 224-237, Dec. 2004.

[12] M. Papageorgiou, *Applications of Automatic Control Concepts to Traffic Flow Modeling, and Control*, New York: Springer-Verlag, 1983.

[13] Y.-S. Huang, Y.-S. Weng, and M. Zhou, "Critical Scenarios, and Their Identification in Parallel Railroad Level Crossing Traffic Control Systems," in *IEEE Trans. Intel. Transp. Syst.*, Vol. 11, No. 4, pp. 968-977, Dec. 2010

[14] L. Qi, M. Zhou, and Z. Ding, "Real-Time Traffic Camera-Light Control Systems for Intersections Subject to Accidents: A Petri Net Approach," in *2013 IEEE*

International Conference on Systems, Man, and Cybernetics, pp. 1069-1074, Manchester, UK, October 13-16, 2013.

[15] A. Tzes, S. Kim, and W. R. McShane, "Applications of Petri Networks to Transportation Network Modeling," in *IEEE Trans. Veh. Tech.*, Vol. 45, No. 2, pp. 391-400, May 1996.